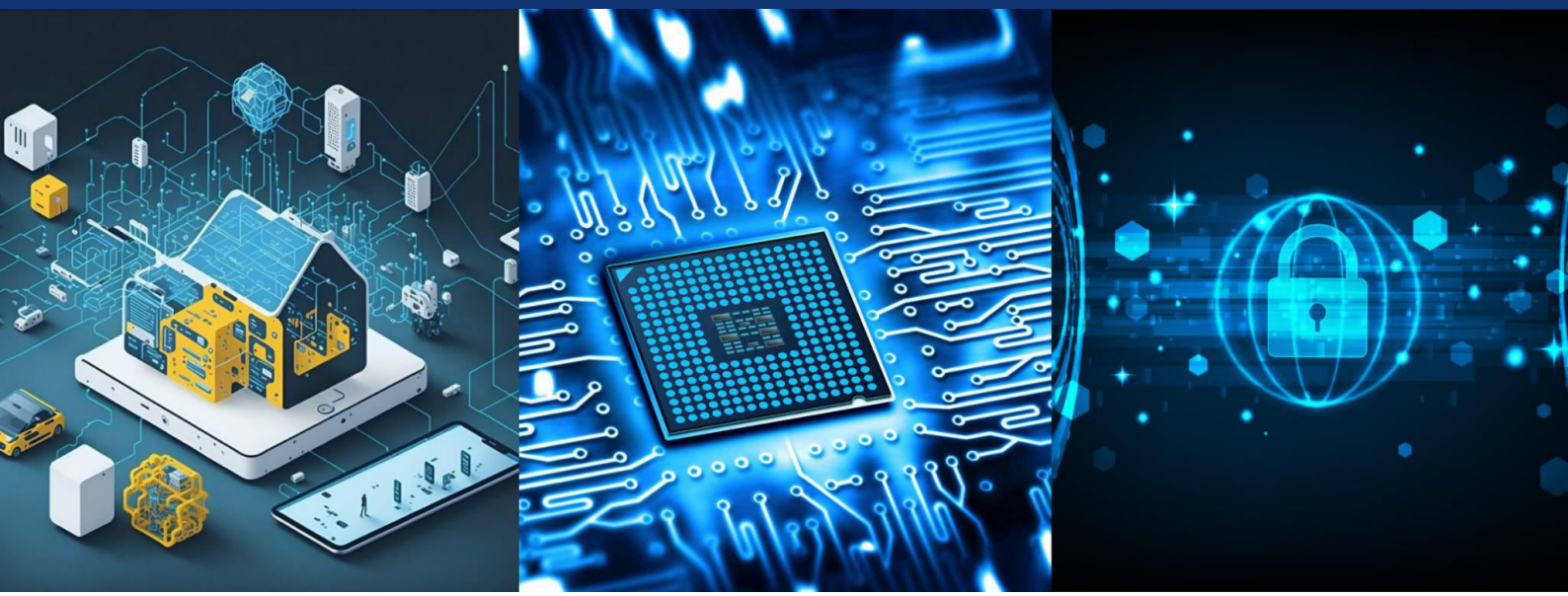
IJIRCCE

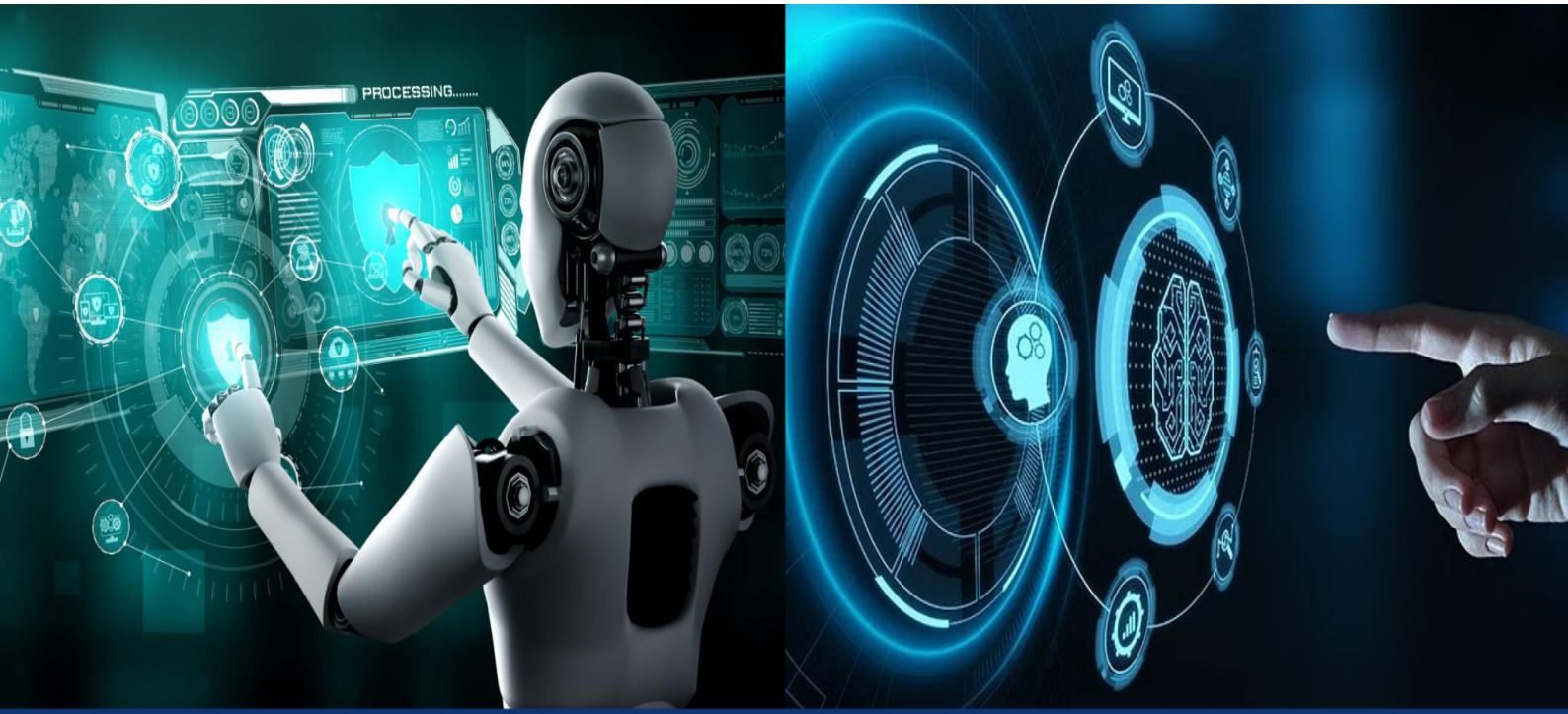
ISSN(O): 2320-9801

ISSN(P): 2320-9798



International Journal of Innovative Research in Computer and Communication Engineering

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)



Impact Factor: 9.927

Volume 14, Issue 10, October 2026

© 2026 IJIRCCE | Volume 14, Issue 10, October 2026 DOI: 10.15680/IJIRCCE.2026.1410048

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 9.927| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

PairMind – AI Augmented Collaborative Coding Workspace

Vardaan Arora[1], Soumya Sinha[1], Shreya Kumari[1], Syed Saifuddin[1], Prof. Josna O.[2]

UG Students, Dept. of Computer Science and Engineering, Rajarajeshwari College of Engineering, Bengaluru, Karnataka, India[1]

Assistant Professor, Dept. of Computer Science and Engineering Rajarajeshwari College of Engineering, Bengaluru, Karnataka, India[2]

ABSTRACT: Collaborative programming can become inefficient when developers must move between separate tools for editing, discussion, debugging, and project tracking. PairMind is proposed as a shared workspace that brings these activities together. It supports collaborative editing, live cursor and participant updates, role-based permissions, messaging, and audio/video communication. The application uses React and Vite for the interface, Flask for backend services, SQLite for data storage, and Socket.IO for real-time events. Monaco Editor provides the coding interface, while the Gemini API supports code explanation, debugging, and improvement suggestions. Additional features include code execution, GitHub integration, and project-management tools. PairMind aims to make team-based programming more convenient, although its synchronization, security, and scalability require further testing.

KEYWORDS: Collaborative coding, real-time sync, Artificial Intelligence (AI), Socket.IO, RBAC (Role Based Access Control), Monaco Editor, AI-based coding, code execution, WebRTC, GitHub.

I. INTRODUCTION

Collaborative software development involves multiple developers working on shared code, files, and project goals. In many workflows, coding, communication, debugging, and task tracking take place in separate applications. Moving between these tools can interrupt work and make it harder for team members to follow changes made by others. Teams may also face delays when sharing code, discussing errors, or coordinating responsibilities.

PairMind is designed as a web-based workspace where developers can work on shared files and communicate without repeatedly switching applications. Team members can edit code, view updates from other participants, run supported programs, and discuss problems through text or audio/video communication. The platform also includes role-based access control and an AI assistant intended to help users understand and improve code.

The proposed implementation combines React and Vite for the frontend, Flask for backend processing, SQLite for persistent data, and Socket.IO for real-time events. Monaco Editor is used as the code editor, and the Gemini API provides AI-assisted programming support. Other planned or included capabilities cover code execution, GitHub repository operations, task tracking, milestones, and documentation generation. The central aim of PairMind is to bring common collaborative programming activities into one workspace and make it easier for team members to coordinate their work.

The main contributions of the project are:

- Allow multiple team members to edit shared code within the same workspace.
- Display cursor positions and participant activity to help users follow ongoing work.
- Apply role-based permissions for Owners, Editors, and Viewers.
- Provide AI-supported code explanation, debugging assistance, and improvement suggestions.
- Allow users to run supported code and inspect the resulting output or errors.
- Provide text messaging and audio/video communication during coding sessions.
- Connect with GitHub for repository access and code commits.
- Support task planning, milestone tracking, and project documentation.

© 2026 IJIRCCE | Volume 14, Issue 10, October 2026 DOI: 10.15680/IJIRCCE.2026.1410048

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 9.927| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

II. SYSTEM OVERVIEW AND ARCHITECTURE

The proposed PairMind architecture contains a frontend, backend, database, real-time communication layer, and integrations with external services. React and Vite form the user interface, while Flask handles application logic and API requests. SQLite stores application data, Socket.IO carries collaboration events, Monaco Editor provides the coding interface, and the Gemini API supplies AI assistance. Other modules support permissions, code execution, WebRTC communication, and GitHub operations.

A. Frontend Module

The frontend allows users to open projects, edit files, communicate with teammates, and organize tasks. React and Vite are used to build the interface, and Monaco Editor provides the code-editing area. API requests connect the frontend to the Flask backend, while Socket.IO messages update code views, cursor positions, and participant status.

B. Backend Module

Flask manages authentication, projects, files, permissions, and database operations. Flask-SocketIO handles real-time collaboration events. Separate service components support AI requests, code execution, GitHub interactions, and project-management functions.

C. Real-Time Collaboration Module

Socket.IO carries updates about code changes, cursor positions, and user activity between participants in a shared room. Before accepting an edit, the server checks whether the user has permission to modify the relevant resource. The proposed PairMind architecture contains a frontend, backend, database, real-time communication layer, and integrations with external services. React and Vite form the user interface, while Flask handles application logic and API requests. SQLite stores application data, Socket.IO carries collaboration events, Monaco Editor provides the coding interface, and the Gemini API supplies AI assistance. Other modules support permissions, code execution, WebRTC communication, and GitHub operations.

D. Frontend Module

The frontend allows users to open projects, edit files, communicate with teammates, and organize tasks. React and Vite are used to build the interface, and Monaco Editor provides the code-editing area. API requests connect the frontend to the Flask backend, while Socket.IO messages update code views, cursor positions, and participant status.

E. Backend Module

Flask manages authentication, projects, files, permissions, and database operations. Flask-SocketIO handles real-time collaboration events. Separate service components support AI requests, code execution, GitHub interactions, and project-management functions.

F. Real-Time Collaboration Module

Socket.IO carries updates about code changes, cursor positions, and user activity between participants in a shared room. Before accepting an edit, the server checks whether the user has permission to modify the relevant resource.

III. METHODOLOGY

PairMind is developed as a web application using React, Flask, SQLite, and Socket.IO. After signing in, users can create or join a workspace and edit shared files through Monaco Editor. Socket.IO distributes code and cursor updates, while JWT authentication and role-based access checks govern access to protected operations.

The Gemini API is used to provide code explanations and suggestions for debugging or improvement. Users can run supported programs and inspect their results, communicate through WebRTC audio/video sessions, and use GitHub integration to manage repository changes. Testing should cover core functionality, concurrent editing, permissions, and performance under multiple-user workloads.

A. User Authentication

JWTs are used to authenticate users and protect project resources. Passwords should be stored as hashes rather than as plain text. After login, the backend validates the token before allowing access to a protected project or collaboration room.

B. Real-Time Code Synchronization

When a participant edits a file, the frontend sends the relevant update to the backend. The backend checks the participant's write permission and processes the change before broadcasting the update to other users in the same workspace. Concurrent editing and recovery from dropped connections should be tested to verify that changes remain consistent.

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 9.927| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

C. Role-Based Access Control

RBAC assigns permissions according to three roles. The Owner manages project settings and team membership, an Editor can modify files within the permissions granted, and a Viewer can inspect shared content without changing it. These rules should be enforced by the backend for every restricted action.

D. AI-Assisted Code Analysis

The Gemini-powered assistant analyzes code and returns possible explanations, fixes, or improvements. Users can inspect the recommendations and apply selected changes in the editor. Since generated suggestions may be incomplete or incorrect, the code should be reviewed and tested before use.

E. WebRTC Communication

WebRTC provides the media connection for audio and video calls, while Socket.IO can coordinate the signaling needed to establish those connections. Participants can control their microphone and camera while collaborating in the workspace.

IV. SYSTEM WORKFLOW AND ALGORITHM

The main workflow of the proposed platform is as follows:

1. Users create an account or sign in with their credentials.
2. A user creates a project or joins a workspace that already exists.
3. The project Owner assigns team roles, such as Owner, Editor, or Viewer.
4. Participants open shared files in Monaco Editor and write or revise code.
5. Socket.IO distributes code updates, cursor positions, and participant activity.
6. The backend verifies permissions before processing protected actions.
7. Users request code explanations, debugging help, or improvement ideas from the Gemini assistant.
8. Users run supported programs and review output or error messages.
9. Team members communicate through text chat or WebRTC audio/video calls.
10. Users assign tasks, monitor milestones, and maintain project documentation.
11. Authorized team members review files and commit changes to connected GitHub repositories.
12. The platform should be assessed through functional, synchronization, access-control, and performance tests.

V. RESULTS AND DISCUSSION

PairMind is intended to provide a common environment for collaborative coding, AI-assisted programming, communication, and project coordination. By bringing these activities together, it aims to reduce the need to switch between separate applications and make team work easier to follow.

PairMind will be evaluated using the following standards:

1. Real-time collaboration: verify that code edits, cursor positions, and participant status are reflected correctly for all connected users.
2. AI assistant: assess whether the assistant explains code clearly, identifies potential errors, and provides useful improvement suggestions.
3. Access control: confirm that Owners, Editors, and Viewers can perform only the actions permitted by their roles.
4. Code execution: test supported languages and check whether the displayed results match the program output.
5. Communication: measure whether chat and audio/video features work reliably during collaborative sessions.
6. GitHub integration: verify repository selection, file access, authentication, and commit operations.
7. Performance and reliability: observe application behavior when several users work at the same time.
8. The evaluation results should identify which functions work as intended and where further development or testing is needed.

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 9.927| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

VI. CONCLUSION AND FUTURE WORK

PairMind proposes a single web workspace for real-time code editing, AI-assisted programming, communication, and project tracking. React, Flask, SQLite, Socket.IO, Monaco Editor, and the Gemini API form the core technology stack. Role-based access control supports permission management, WebRTC enables audio/video collaboration, and GitHub integration supports repository workflows. The platform's usefulness will depend on validating synchronization, security, execution safety, and performance through systematic testing. Although PairMind combines live code sharing, AI assistance, and project-management tools, it has limitations. Network instability can delay synchronization, and simultaneous edits may expose consistency issues. Gemini-generated suggestions can be inaccurate and must be reviewed. Audio/video quality depends on network conditions, while executing untrusted code requires strong isolation and security controls. SQLite may also become a bottleneck as concurrent usage grows. Future work can improve synchronization and recovery, strengthen sandboxing for code execution, refine AI-assisted debugging, and evaluate a more scalable database.

REFERENCES

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural ma chine translation by jointly learning to alignand translate. arXiv preprint arXiv:1409.0473 (2014).
[2] Nghi DQ Bui, Yijun Yu, and Lingxiao Jiang. 2021. Self-Supervised Contrastive Learning for Code Retrieval and Summarization via Semantic-Preserving Trans formations. In Proceedings of the 44th International ACM S.
[3] WenchaoGu,ZongjieLi,CuiyunGao,ChaozhengWang,HongyuZhang,Zenglin Xu, and Michael R Lyu. 2021. CRaDLe: Deep code retrieval based on semantic.
[4] Xiaodong Gu, HongyuZhang, and SunghunKim.2018. Deepcode search. In 2018 IEEE/ACM 40th International Conference on Software Engineering (ICSE). IEEE, 933–944.
[5] Sonia Haiduc, Gabriele Bavota, Andrian Marcus, Rocco Oliveto, Andrea De Lucia, and Tim Menzies. 2013. Automatic query reformulations for text retrieval in soft ware engineering. In 2013 35th International Conference on Software Engineering (ICSE). IEEE, 842–851.
[6] Fei Lv, Hongyu Zhang, Jian-guang Lou, Shaowei Wang, Dongmei Zhang, and Jianjun Zhao. 2015. Codehow: Effective code search based on api understanding and extended boolean model.
[7] LingXu,HuanhuanYang,ChaoLiu,JianhangShuai,MengYan,YanLei,andZhou Xu. 2021. Two-Stage Attention-Based Model for Code Search with Textual and Structural Features. In 2021 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER). IEEE, 342–353.
[8] Wei Ye, Rui Xie, Jinglei Zhang, Tianxiang Hu, Xiaoyin Wang, and Shikun Zhang. 2020. Leveraging code generation to improve code retrieval and summarization via dual learning. In Proceedings of The Web Conference 2020. 2309–2319.
[9] Ziyu Yao, Jayavardhan Reddy Peddamail, and Huan Sun. 2019. Coacor: Code annotation for code retrieval with reinforcement learning. In The World Wide Web Conference. 2203–2214.
[10] Xin Xia, Lingfeng Bao, David Lo, Pavneet Singh Kochhar, Ahmed E Hassan, and Zhenchang Xing. 2017. What do developers search for on the web? Empirical Software Engineering 22, 6 (2017), 3149–3185.



INTERNATIONAL STANDARD SERIAL NUMBER INDIA

INNO SPACE SJIF Scientific Journal Impact Factor

doi crossref

निस्केयर NISCAIR

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN COMPUTER & COMMUNICATION ENGINEERING

9940 572 462 6381 907 438 ijircce@gmail.com

www.ijircce.com

Scan to save the contact details